Software Surprise

Three Invisible Problems of Weapon System Software Development

LT. COL. L. D. ALFORD, U.S. AIR FORCE

With technology advancing at a rapid pace, yesterday's state-of-the-art software is outdated today. If this were the only problem facing the development of DoD weapon systems software, it would be enough. But more problems lie ahead—at least three other critical software issues cause major problems for program managers, testers, and ultimately customers.

These three problems are software-induced workload, software system complexity, and software systems costs. Even though these three problems have an enormous impact on the overall system, they are given little visibility because program managers rarely realize they exist.

All three of these problems are program-invisible. What I mean by this, is that they are rarely tested or even thought about until after they have become a serious difficulty for the program. The dilemma is that these software/integration problems are one of the foremost reasons for customer dissatisfaction and increased systems costs.

Software-Induced Workload

Software-induced workload is what a program is attempting to reduce or avoid by adding software to the system. With the complexity of current hardware systems and the missions they support, software is used, primarily, to integrate and consolidate systems so the equipment operators can accomplish the mission with decreased workload and increased mission effectiveness. The only problem is, no one has discovered a way to measure workload.

Specifically, all the measures we currently have for workload are qualitative and not quantitative. In the past, engineers tried to use quantitative measures such as altitude and airspeed capture to measure workload. Unfortunately, these measures have nothing to do with workload. For instance, using a digital altimeter, a test pilot can fly an aircraft 10 feet. The workload is extremely high and even a test pilot can't accomplish this task for long, but according to engineering measures, the workload would not be that great because the event can be achieved. This train of engineering analysis resulted in the tape altimeters on the C-5, C-141, and F/FB-111 aircraft. Aviators who have flown these aircraft will testify to their "low" workload after they have become proficient in the systems; however, analytical tests with other aviators always prove them wrong.

In spite of this, because there is no usable measure for workload, when we try to measure workload, data from such analyses are always suspect: the sample size is rarely large, the statistical confi-

Alford is the Chief of the Test & Evaluation Division, Special Operations Forces, System Program Office. He is a graduate of APMC 98-2, DSMC.

dence is low, and no method exists to quantitatively measure the workload. What this means is, when we try to evaluate whether, for instance, we want to reduce the number of crewmembers in the cockpit, our decision is not based on analysis and tests, but rather a hope based on politics and cost of the additional crewmembers.

The best examples of this are the MC-130H and the current Air Force glass cockpits and heads-up displays (HUD). The MC-130H is one of the best-missionized aircraft in the world. The pilot puts the cue on the dot and can fly any terrain-following profile programmed by the navigator and the aircraft system. On the other hand, it is a poor instrument aircraft. The tape digital displays make it extremely difficult to fly.

In like fashion, the glass cockpits and HUDs of Air Force aircraft are based on similar tape displays. These displays are great for civil aircraft, which are flown literally from takeoff to touchdown on autopilot, but become burdensome "workload sinks" for military tactical flights. This workload problem will continue to be an obstacle until we discover a method to quantitatively measure workload. Fortunately, research toward this end is ongoing, but a majority of fielded and future systems have been or are being designed without any clue to the workload involved.

Another example is radio frequency changes in aircraft that use digital integrated radio systems. Changing a frequency using the old analog dial paradigm is relatively simple. The pilot inputs the frequency by turning a dial on the console. In a software display, the pilot must first find the page for frequency entry, then select the proper place for the entry, and finally, input the digits from a touch-pad. This is at least 10 times greater workload than the analog dialing system, yet the new paradigm appears to demand it. Multiply this example times the number of system inputs the pilot must make to accomplish any mission. These examples have just touched the periphery of the problems associated with workload. Suffice it to say that software/integrated systems generally have significantly increased workload without a proportional increase in mission effectiveness.

Software Complexity

Software complexity is the second great hidden problem in software development. Because software affects so many systems and is so intrusive, it has become impossible to fully test even the safety-related effects of the software.

When a new software build is installed in an aircraft, unknowns are rampant, and the "bugs" are rarely fully discovered even during flight tests. Some problems lie dormant until the systems are well deployed.

One example was an Operational Flight Program (OFP) release on the MC-130H. This release was supposed to affect only the terrain-following system of the aircraft. The aircraft was released for flight under the assumption that it was okay as long as the terrain-following system was not engaged. In the middle of a training flight, during an engine-out approach, the crew noticed that the "ball" (primary flight coordination instrument) was indicating the opposite of the correct direction. If this OFP had made it into the fleet, or a test crew had not been flying the aircraft, in all likelihood a smoking hole would have appeared where a multimillion-dollar aircraft had once been. Although this example may appear extreme, hundreds of others, in and out of flight tests, abound. Software/integrated systems increase this risk, and the risk is proportional to increasing code and increasing integration complexity.

The C-21 (Lear 35) is another example. In this aircraft, if an oil pressure circuit breaker was pulled/popped, certain engine control settings would result in a fire light on an engine. An operational crew discovered this problem. Because of it, they shut down a good engine and landed short of their destination. They happened to get two fire lights, one on each engine. Luckily, they realized the indicating system was the source of the problem before they shut down both engines. The circuit breaker had popped due to a faulty circuit problem. A sneak circuit caused the fire warning in the indicating system. The crew and passengers were placed at risk due to the malfunction of a $10 piece of equipment. This has been fixed since the incident, but who knows how many other similar problems wait to be found? Software and integration complexity increases risk.

Software Systems Costs

The third problem is related to the first two. Software always requires future improvements and rewrites. Complex software invariably comes with "bugs," and the "bugs" are never entirely discovered. Modifications and fixes add their own "bugs" resulting in future modifications and fixes.

Because of software integration and complexity, the cost of fixes, modifications, and improvements is high. Rarely are software systems provided with sufficient life-cycle funding for these fixes, improvements, and modifications. Software has become so intrusive that the simplest components, on many aircraft, incorporate some software. In fact, even such things as the clocks, circuit breakers, and pressurization systems in most modern aircraft incorporate or are dependent on software for correct indication and operation. Most aircraft are now to some degree fly-by-wire and engine control-by-wire. This trend in controls and systems shows no sign of decrease or change.

Funding must be provided for any software system until the decommission of the system. This is a given that most services and program offices have yet to acknowledge. For example, numerous electronic warfare systems are not adequately funded for software changes, yet are currently going through major changes. This has resulted in serious program problems such as multiple OFPs in multiple versions being accomplished by more than one agency. The resulting costs are much more than they would have been if software changes had been programmed for the life of the system. The examples of the MC-130H and the

C-21 resulted in cost increases, which were not planned and which could have radically affected the safety of the aircraft if the funding had not been made available.

Lessons Learned Simple, Solutions Complex

The lessons to learn from these three invisible software/integration problems are simple. Their solutions are not. First, try to evaluate workload when developing a system. Attempt to use nonintegrated systems when possible and especially when workload studies indicate a problem. The DoD must fund research and development to discover effective quantitative workload measures. Second, plan and test for as much as possible and be ready, during all program phases, for software problems to "rear their ugly heads." Do not be content with minimal software testing even when risk is low. Finally, fund software for the life of the system.

These three issues, software-induced workload, software system complexity, and software systems costs are critical, rarely visible program problems. They should be primary considerations during all program phases. They may be invisible now, but unless tamed, they will drive your program and the capability of your weapon system.

DoD Announces TOP 100

On Feb. 4 the Department of Defense announced that the fiscal year 1998 report of "100 Companies Receiving the Largest Dollar Volume of Prime Contract Awards (Top 100)" is now available. To read or download this report or other DoD contract statistics, go to http://web1.whs.osd.mil/diorhome.htm on the World Wide Web.

DOT&E RELEASES ANNUAL REPORT

The Department of Defense Director of Operational Test and Evaluation, Philip E. Coyle, announced Feb. 11 the release of his 1998 Annual Report to the Congress and the Secretary of Defense. The report describes the operational and live-fire testing performed on 160 military systems in 1998 and provides an assessment of the contribution each weapon system makes to *Joint Vision 2010*, the conceptual framework for how U.S. forces will fight in the future.

The report reviews the state of test and evaluation capability within the Department and makes recommendations for investment at major test and training ranges. The report is at http://www.dote.osd.mil on the World Wide Web.